A FLEXIBLE FRAMEWORK FOR A WEB TRANSACTIONAL APPLICATION

[1]Mohd Khanapi Abd Ghani, [2]Abdul Samad Shibghatullah, [3]Mohamad Hisyam Selamat
[1]Faculty of Engineering and Computing, Coventry University,
West Midlands, CV1 5FB, UK. abdghanm@coventry.ac.uk
[2]School of Information System, Computing and Mathematics
Brunel University, Uxbridge, UB8 3PH, UK. cspgass@brunel.ac.uk
[3]Fakulti Perakaunan, Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia. hisyam@uum.edu.my

Abstract. *Software development project becomes difficult because of the complexity in the business requirements, rigid framework and unpredictable performance. These cause difficulties to deliver the software on time, to maintain it and to adapt to new requirements. This research proposes a Web-Based Transactional Application Architecture Framework (WTBF) to simplify the development of a complex software project by using template approach that contains assembled pieces of reusable software components. The framework provides a generalized component-based architectural template that can be reused by the users (software developer and an independent software vendor) to develop multi-tier transactional applications within a specific domain or for a new software project. One of its significant benefits is that users can concentrate on their business components development tasks instead of spending time on redeveloping basic software infrastructure from the scratch. The WTBF framework is developed by using the Java 2 Technology and the Model View Controller (MVC) is used for the architectural model.*

1.0 Introduction

Managing software development projects is more difficult in a complex business environment, rigid design approach and due to unpredictable software performance. This results in late delivery, complicated systems maintenance and inability to adopt to new requirements. A possible solution to this is that a flexible framework that can shorten software development period, ease systems maintenance and adapt to new requirements.

This paper proposes a Web-Based Transactional Application Architecture Framework (WTBF) to simplify software development projects by using template approach that contains assembled pieces of reusable software components. The framework provides a generalized component-based architectural template that can be reused by the users to develop multiple-tiers transactional applications within a specific domain or for a new software project.

One of the significant benefits of WTBF is that users can concentrate on developing business components rather than spending time on developing the basic software infrastructure from the beginning. WTBF can also reduce the software development cost and period; and indirectly can optimize task distribution amongst the project team members.

Before describing WTBF, this paper presents some of the problems that are related to software development project. Then, the issues that highlight the need for a WTBF are discussed. Thereafter, the component of WTBF is defined and described in detail. Finally, the conclusion and suggestions for further research are dealt with.

2.0 Software development project problems

Some of the significant problems to software development project are late delivery, just-do-it approach, poor system maintenance and unpredictable performances.

Late delivery will create other problems such as, increased development cost, bad reputation and stress to staffs. The reasons that contribute to late delivery are such as, failure to understand requirements, poor planning, and mismanagement. Almost new software project starts from the scratch whereby every single component and service has to go through a proper software development process. This is due to inadequate reuse of software components. As a result, longer development period is involved and in turn increases the project cost.

On the other hand, just-do-it approach will produce a software system that is brittle and inflexible because no consideration is given to the need to adapt to new requirements and maintenance efforts. A proper software

development methodology is important as it will ensure that every aspect of the systems development are undertaken exhaustively.

The difficulty to maintain a system is caused by poor documentation, and failure to understand and identify beforehand how the modification of the software could be made. Thus, software modification or enhancement will take a longer period to accomplish.

Unpredictable performance occurs when the delivered software cannot meet the response time guidelines and criteria. This leads to problems in managing system performance and addressing the performance issues. The same code may be redone over and over again leading to unpredictable performance.

With regard to the above issues and a fierce market competition, companies do not have other choice except to change. They must constantly oblige to adapt to new needs, make their products evolve, change their development process and establish new collaborations (Crnkovic, 2001). These evolutions require the implementation of a flexible application systems framework architecture that is adaptable to the evolution of business products and business process.

3.0 Issues related to WTBF

WTBF is a web-based, model-view-controller, and reuse approach. The reasons are explained in details below:

3.1 Software technology evolution

In the early decades of the computer existence, mainframe technology was the main tool for the computing system and all software were tied to this central entity. The applications in this type of architectural association are commonly referred to as *single tier* applications (Ahmed et. al., 2001). From the application perspective, single tier is the most problematic one because it has to face the changing technology and increase business requirements in the future.

The second generation of the application technology that is the client-server (*two-tier*) approach mitigates the above issues by moving the presentation aspects and some of the business logic to a separate tier. The web applications involved multiple tiers (*n-tier*), either by physical or logical separation that is according to responsibilities and services. The n-tier approach attempts to achieve a better balance by separating the presentation logic from the business logic and the business logic from the underlying data.

Based on the fact that the n-tier architecture involved multiple layers and component services, it is important to note that a proper and systematic architecture framework needs to be designed and developed at the beginning stage of the project. The software developers realize that getting the right architecture framework is a critical success factor for the software design and development project.

3.2 Software Architecture

Software architecture is a principal mechanism and pattern that defines and communicates the structure of a system (Jacobson et. al., 1998). The architecture allows applications and components to evolve gracefully and enables software developers to achieve significant levels of reuse, enables components to work well together, and allows software developers to develop under stable systems that are easier to maintain.

Local software developers (in Malaysia) usually do not pay any attention to the architectural elements of their business to create better products and to enable them to compete in the global market (Mohamed M., 2005). The traditional approach such as structured sequential application development methodology (SSADM) is still used in their software projects. This methodology has often been criticized for lack of modularity, reusability, maintainability and compositionality.

Nowadays, software developers must pay more attention to the architectural or design element of their software product. Similar to how the architecture helps the construction or house developer to create his masterpieces, the same approach can be applied to software development. There are many dimensions in the software architecture that is from the platform and development to the application.

Model-view-controller (MVC) is a software architecture that separates an application's business logic, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others. It separates objects into one of three categories — **models** for maintaining business logic, **views** for displaying all or a portion of the data, and **controllers** for handling events that affect the model or view(s). By applying the MVC architecture to an application, the developer can separate core business model functionality from the presentation and control logic that uses this functionality. Such separation allows multiple views to share the same enterprise data model, which makes supporting multiple clients become easy to implement, test, and maintain.

3.3 Template and reuse approach

According to Cambridge Dictionary (2003), template is a pattern made of metal, plastic or paper, which is used to make many copies of a shape or to help cut material accurately. In the context of software, it is a system that helps us arrange information on a computer screen and logic.

A general understanding of template is that it helps us to develop products faster and more accurate whilst avoid major risks of erroneous mistakes. The existing products or patterns or software framework will be reused and customized in order to match with the targeted products that are going to be developed. Reusing the experienced products that are the product that have been used by many users, would be less risky compared to the product that is developed from the scratch.

Many companies realize that reuse architecture is the most effective path towards dramatic improvement of both development costs and period. According to Mcllroy (1996), "Software reuse is simple as develop systems of components of a reasonable size and reuse them". In this case, the idea of "component systems" is extended from merely based on code to requirements, analysis models, design and test. All the stages in the software development process are subject to "reuse".

By using reuse approach, the developers can save problem-solving effort throughout the development chain. In addition, they can minimize the redundant work and enhance the reliability of their work because each of the reused component system has already been reviewed and inspected according to its original development.

The experienced companies such as, Hewlett-Packard, Microsoft, AT&T, Motorola, IBM and Toshiba gained a significant cost and time saving from the systematic reuse. They have obtained around 90% reuse levels in certain projects or areas and gained a substantial result in the following elements:

Time to market: reductions of 2 to 5 times
Defect density: reductions of 5 to 10 times
Maintenance cost: reductions of 5 to 10 times
Overall software development cost: reductions of around 15% to as much as 75% for long term projects.

Source: Jacobson et. al. (1998); Yourdan (1996) ; and Cusumano(1991)

Many companies are trying to improve their software development performance. Therefore, the template and software reuse must become key parts of their software engineering strategy. The above discussion shows that a giant software company such as Microsoft and IBM gained success from the software reuse effort. Hence, software reuse is a vital strategy and approach for the viability of the software industry.

4.0 The WTBF framework

In this paper, we propose a flexible framework that supports a web-based system, MVC approach and can accommodate reusable component. This relationship is illustrated in figure 1. The framework consists of three main components that are message grammar format, front end and back end. Message grammar format is a new format that is proposed by this framework to deal with data transmitting. This new format can transmit data quicker than the existing format. Front end components consist of Java Web Start, Java Beans and HSQLDB database. Back end component uses Java 2 Platform, Enterprise Edition (J2EE).

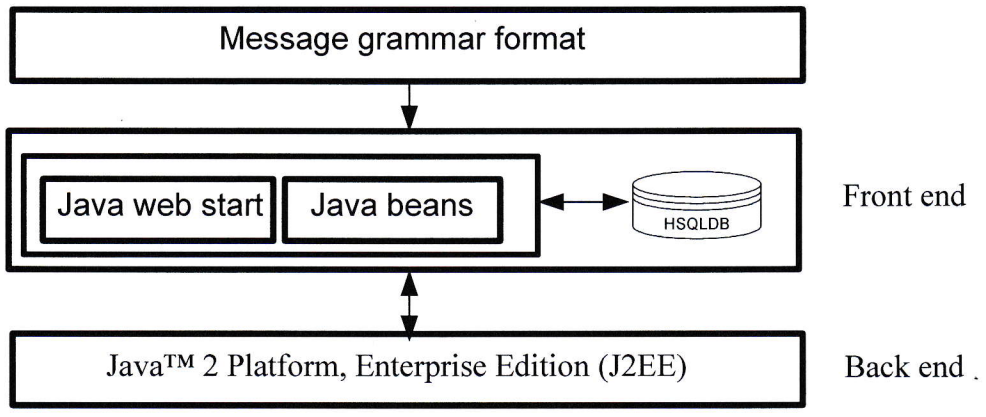

Figure 1: The WTBF framework components

4.1 Choice of front end technology

In the modern applications, that involve business logics and requirement of a complex user interface, web application requires systematic architecture framework in order to be beneficial for client computers to share in the execution of some of the business logics. The web application executes business logic on the server only; therefore, the client browser which is just a simple text cannot last longer. Therefore, a thick client/web-centric, which uses Java Web Start and Java Beans technology, is chosen and integrated in the WTBF front-end architecture. The Java Web Start overcomes user interface and connectivity limitations and offers client-side software distribution features as well.

Java Beans technology is software components for the Java 2 Platform. It is built on the Java Foundation Classes API (JFC). JFC is an infrastructure that enables developers to build components, each in the language of their own choice, and share these objects with one another in order to build a bigger and more complex system. One of the key features that make WTBF front end architecture suitable for the web-centric is its components that enable automatic download when necessary. Normally, when utilising Java Network Launching Protocol (JNLP) on the client, the application control is cached in a 'jar file'

Table 1 compares the different technologies that are used for designing Web clients. These different factors influence the design of the WTBF front end architecture (http://java.sun.com/developer/technicalArticles/JavaLP/jawawebstart/).

Table 1: Comparison of different technologies used for designing Web clients

Factors	Applets	XML/HTML-based clients	Java Web Start
User interface	Moderate to sophisticated	Simple to moderate	Moderate to sophisticated
Offline support	No	No	Yes
UI response	Network independent	Network independent	Network independent
Interactively	Browser limited	Browser/markup limited	Open
First use response	Minutes	Seconds	Minutes
Subsequent use response	Minutes	Seconds	Seconds
Bandwidth usage	Variable	Fixed	Flexible
Lightweight client support	Limited	Open	Limited

As we can see, while applets take same time to download in the first use and each subsequent use, Java Web Start applications will load faster on the subsequent uses due to the product's inherent caching features.

4.1.1 Hypersonic Database (HSQLDB)

Hsqldb is a relational database engine written in Java. WTBF client use Hsqldb for the offline transaction whereby the transactional messages are stored locally. The transaction is written in the disk instead of memory. This is to ensure that the reliability and the stability of the transaction data in local disk are maintained. Furthermore, the response time to save and retrieve the data is faster than from the server.

4.2 Choice of Back End Application Patterns

WTBF back end architecture adopt the Model View Controller (MVC) architecture pattern. The main reason behind this is that to minimize the coupling among objects in a system by aligning them with a specific set of responsibilities in the area of the persistent data and associated rules, presentation, and the application logic.

The Java™ 2 Platform, Enterprise Edition (J2EE) has been selected for building server-side and enterprise-class application for WTBF. The enterprise software development is a complex task and requires extensive knowledge of many different areas. For instance, a typical enterprise application development effort requires us to be familiar with the inter-process communication issues, security issues, database specific access queries, and others. J2EE uses build-in concept and transparent to support those processes and similar services. As a result, the developers are able to focus on implementing business logic codes rather than codes that support basic application infrastructure.

The J2EE enterprise development model also encourages a demarcation between system development, deployment, and execution. Because of this, the developers can defer deployment details such as, the actual database name and location, and can host specific configuration properties to the deployer.

4.3. Design Messages Grammar Format

The message contains object data about the transactional posts by the user. A message is categorized into two components namely the message header and message detail. Each message is constructed by special characters that is coined as delimiters. They are the segment terminator, the segment separator, the field separator, the value separator and the repetition separator. The system should understand the grammar of the messages before it can be read and sent for further processes in the back end component.

In the absence of other considerations, it was suggested that the delimiter values are designed as shown in Table 2.

4.3.1 MsgHdr

The message header segment defines the intent, source, destination, and some specifics of the syntax of a message. The syntax is controlled by the real-world event where it creates the need for data to flow among systems. The real-world event is termed as the transaction code.

4.3.2 MsgDetail

The message detail contains detailed data attributes about the transaction. These are the actual data to be stored in the persistence storage. The data are controlled by the field separator, the value separator, the repetition separator and the segment terminator.

Table 2: Message Component Delimiter Values

Delimiter	Suggested Value	Encoding Char Position	Usage
Segment ID Separator	¤	1	Separates between the message header and the message detail.
Value/Component Separator	¥	2	Separate between field name and its value
Repetition Separator	,	3	Separates multiple occurrences of a field.
Field Separator	¶	4	Separates two adjacent data fields within a segment. It also separates the segment ID from the first data field in each segment.
Segment Terminator	ÿ	-	Terminates a segment record. This value cannot be changed by implementers.

5.0 Conclusions

This paper has described the software development project problem in the general. The problems will leads to late delivery, hard to maintain and inflexible to new requirements. This paper proposes a flexible framework called WTBF to deal with those problems. The aim of WTBF is to simplify the development of complex software projects by using template approach. Another significant contribution of WTBF is a new format for the message transaction that will expedite the data transmission between server and client. WTBF uses Java Web Start and JavaBeans in the front end part and back end part uses J2EE.

J2EE is the Sun's enterprise Java solution. It is the standard for developing multi-tier enterprise applications. By using J2EE, WTBF provide a seamless and natural route into the JavaBeans, Java Web Starts, Java servlets, Java Server Pages, and XML. Thus J2EE provides the bedrock upon which the technology is build: a stable, well-supported, industrial standard platform, which, via the whole range of Java APIs, provides easy and robust access to almost every application technology imaginable.

A future development will be to apply the truly component objects on the WTBF back end components engine. For example, the business logic should provide simple interface to integrate with the existing business logic or for new business logic. The design of the message components should enable software developers to adapt to future message format standard such as, Extended Markup Language (XML). The existing message format that focuses only on the object string will be suitable for Java technology platform. It requires intensive conversion effort when the different technology platform is decided to be used.

6.0 References

Ahmed K.Z., Umrysh C.E., (2001). Developing Enterprise Java Applications With J2ee And UML, Addison-Wesley.

Cambridge Advanced Learner's Dictionary (2003), Cambridge University Press.

Conallen J., (2000). Building Web Applications With UML, Addison-Wesley.

Crnkovic I., (2001). Component-Based Software Engineering – New Challenges in Software Development, Software Focus, Sweden. Vol.2 (4), pages 127-133.

Cusumano M.A. (1991), Japan's Software Factories : A Challenge to US Management. New York : Oxford University Press.

Garlan D, Software Architecture: A Roadmap, School of Computer Science Carnegie Mellon University , garlan@cs.cmu.edu. Downloaded: 23/11/2004

Crnkovic, I., Filipe, J. K., Larsson, M., and Lau, K. 2001. Object-oriented design frameworks: formal specification and some implementation issues. In *Databases and information Systems*, J. Barzdins and A. Caplinskas, Eds. Kluwer Academic Publishers, Norwell, MA, 237-251.

Jacobson I., Griss M., Jonsson P., (1998). Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley.

Kassem N., Enterprise Team., (2000). Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition, Sun Microsystems, Inc.

Martin Fowler, (2004). UML Distilled Third Edition: A Brief Guide to The Standard object modeling language, Addison-Wesley.

Rothernberger M.A., Dooley K.J., Kulkami U.R., (2003). Strategies For Software Reuse – A Principal Components Analysis Of Reuse Practices, IEEE Transactions on Software Engineering. Vol. 29. No.9, September. 2003.

Xiang Y., Gu Q., Li Z., (2003). A Distributed Framework of Web-Based Telemedicine System, Proceedings of the 16th IEEE Symposium on Computer-Based Medical Systems (CBMS'03) 1063-7125/03 © 2003 IEEE

Yourdan E., (1996), Visual Basic 4. Application Development Strategies, VIII(2), 1-16